

Atty. Docket No. MS303532.1

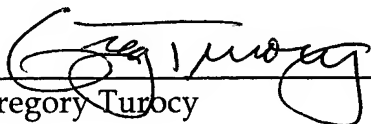
A COST-BENEFIT APPROACH TO AUTOMATICALLY  
COMPOSING ANSWERS TO QUESTIONS BY  
EXTRACTING INFORMATION FROM LARGE  
UNSTRUCTURED CORPORA

by

Eric J. Horvitz, David R. Azari, Susan T. Dumais, and Eric D. Brill

MAIL CERTIFICATION

I hereby certify that the attached patent application (along with any other paper referred to as being attached or enclosed) is being deposited with the United States Postal Service on this date August 6, 2003, in an envelope as "Express Mail Post Office to Addressee" Mailing Label Number EV330020395US addressed to the Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

  
\_\_\_\_\_  
Gregory Turcotte

Title: A COST-BENEFIT APPROACH TO AUTOMATICALLY COMPOSING  
ANSWERS TO QUESTIONS BY EXTRACTING INFORMATION FROM  
LARGE UNSTRUCTURED CORPORA

5

## TECHNICAL FIELD

The present invention relates generally to computer systems, and more particularly to a system and method that facilitates automated extraction of information from a heterogeneous knowledge base by applying a utility model to selectively normalize the knowledge base.

10

## BACKGROUND OF THE INVENTION

Over the years, computer systems designers have pursued the challenge of developing computational architectures that have the ability to generate answers to freely-posed questions. General question-answering systems typically depend on automated processes for analyzing questions and for composing answers from a large corpus of poorly structured information. In recent years, systems have been developed that employ the resources of the Web as a corpus of information for answering questions. Web-based question answering systems typically employ rewriting procedures for converting components of questions into sets of queries posed to search engines, and converting query results received from the search engines into one or more answers.

20

Many text retrieval systems, for example, operate at the level of entire documents. In searching the web, complete web pages or documents can be returned. There has been a recent surge of interest in finer-grained analyses focused on methods for obtaining answers to questions rather than retrieving potentially relevant documents or best-matching passages from queries—tasks information retrieval (IR) systems typically perform. The problem of question answering, however, hinges on applying several key concepts from information retrieval, information extraction, machine learning, and natural language processing (NLP).

25

Automatic question answering from a single, constrained corpus is extremely challenging. Consider the difficulty of gleaning an answer to the question “*Who killed Abraham Lincoln?*” from a source which contains only the text “*John Wilkes Booth altered history with a bullet. He will forever be known as the man who ended Abraham Lincoln’s life.*” As can be appreciated, however, question answering is far easier when the vast resources of the Web are brought to bear, since hundreds of Web pages contain the literal string “*killed Abraham Lincoln.*”

Most approaches to question answering use NLP techniques to augment standard information retrieval techniques. Systems typically identify candidate passages using IR techniques, and then perform more detailed linguistic analyses of the question and matching passages to find specific answers. A variety of linguistic resources (part-of-speech tagging, parsing, named entity extraction, semantic relations, dictionaries, WordNet, *etc.*) can be employed to support question answering.

In contrast to these rich natural language approaches, others have developed question answering systems that attempt to solve the difficult matching and extraction problems by leveraging large amounts of data. In one such system, redundancy provided by the web can be exploited to support question answering. Redundancy, as captured by multiple, differently phrased answer occurrences, facilitates question answering in two key ways. First, the larger the information source, the more likely it is that answers bearing close resemblance to the query can be found. It is quite straightforward to identify the answer to “*Who killed Abraham Lincoln?*” given the text, “*John Wilkes Booth killed Abraham Lincoln in Ford’s theater.*” Second, even when no exact answer can be found, redundancy can facilitate the recognition of answers by enabling procedures to accumulate evidence across multiple matching passages. In order to support redundancy however, a plurality of variously phrased queries may have to be submitted to one or more search engines. This type of approach may place an unacceptable performance burden or load on search engines responding to the query especially considering the number of users that potentially utilize network resources.

## SUMMARY OF THE INVENTION

The following presents a simplified summary of the invention in order to provide a basic understanding of some aspects of the invention. This summary is not an extensive overview of the invention. It is not intended to identify key/critical elements of the invention or to delineate the scope of the invention. Its sole purpose is to present some concepts of the invention in a simplified form as a prelude to the more detailed description that is presented later.

The present invention relates to systems and methods to automatically extract information from large unstructured or semi-structured heterogeneous corpora such as a knowledge base of local and/or remote data stores of information, or the entire World Wide Web. A normalization component employing a statistical model of the likelihood that different information retrieval operations will be valuable and a utility model is provided to characterize the information value of different kinds of retrievals from the knowledge base. The characterizations are used to control in a dynamic manner processes that extract or glean unseen, previously unknown, and/or disassociated information from the knowledge base. For example, the knowledge base may include a plurality of web sites that are interconnected across the Internet.

Questions or queries posed by a user to an interface component (*e.g.*, web service) are automatically reformulated into a determined subset of queries or other types of information-gathering operations, wherein the utility model performs a cost-benefit analysis to control and optimize potential reformulated queries that are submitted to the knowledge base (*e.g.*, one or more search engines accessing databases) in order to obtain answers to the questions. In this manner, unstructured databases having individual components of information are normalized to find/connect the components in order to provide previously unknown information to users such as in the form of an answer to a specifically worded question.

The cost-benefit analysis considers and balances the cost of acquiring new information, *e.g.*, via submitting additional reformulated queries, versus the benefit or value of receiving a more accurate answer to a question posed by the user, by integrating

new information acquired from the acquisition. By dynamically balancing or optimizing these considerations, back-end search processing for answers can be enhanced while achieving desired results when determining answers (*e.g.*, performance of search engines enhanced by processing less queries). Also, preference tools can be provided to enable users to provide input relating to assessments of costs for retrieving information and the value placed on obtaining more accurate answers to questions.

In one particular aspect of the present invention a Web-centric question-answering system is provided although the present invention can be applied to substantially any type of heterogeneous knowledge base. A layer or layers of probabilistic analysis and learning can be applied to guide extraction of information from the Web in a Web-centric question answering system or service, for example. One or more phases of machine learning can be employed to build Bayesian models that predict the likelihood of generating an accurate answer to questions along with coupling such predictive models with considerations of the value and costs of various web-querying actions.

Other aspects of the present invention include extending decision-making considerations to mixed-initiative interaction, wherein decision models consider real-time input from users to refine or reformulate questions (*e.g.*, dialog component to ask users to reformulate questions). Thus, beyond selecting the best web-querying actions to take, the present invention can include cost-benefit analyses that considers when it would be best to ask a user to reformulate a question rather than expending effort on processing a query that may be expensive or likely to yield inaccurate results. In such an analysis, an assessment of the cost of delay and effort associated with a query reformulation and the likelihood that a reformulation would lead to a better result can be considered and modeled.

The predictive power of models of answer accuracy can be enhanced by considering additional features of questions and query rewrites, and extending inference methods to acquire or reason about notions of topic, informational goals, and overall context of a user posing a question. This can include enhancing models for predicting

topic and high-level intentions associated with questions from tagged libraries of questions, posed by users of online encyclopedias and/or other information sources. The models can provide predictions of the high-level information goals, topic, and desired level of detail of users, based on parts of speech and logical forms provided by a Natural Language Processor parse of questions.

Beyond extending probabilistic models of accuracy and expected value analysis, question –answering systems in general can be refined in several ways. Refinements include introducing new variants of query rewrites and modifying methods for combining search results into candidate answers. In addition to guiding real-time question-answering procedures, decision-analytic evaluative and control machinery can serve as a tool or service, enabling systems to probe in an explicit manner the utility of making specific modifications to question-answering systems.

In a broader aspect, the present invention includes methods for introducing a “normalizing layer” of probabilistic analysis of accuracy coupled with utility-guided query control for guiding the extraction of information from the Web and/or other knowledge base, in support of a variety of tasks requiring information synthesis from large, unstructured corpora.

To the accomplishment of the foregoing and related ends, certain illustrative aspects of the invention are described herein in connection with the following description and the annexed drawings. These aspects are indicative of various ways in which the invention may be practiced, all of which are intended to be covered by the present invention. Other advantages and novel features of the invention may become apparent from the following detailed description of the invention when considered in conjunction with the drawings.

#### BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a schematic block diagram of a normalization system employing a utility model for extracting information from an unstructured corpora in accordance with an aspect of the present invention.

Fig. 2 is a diagram of a preference component and utility model in accordance with an aspect of the present invention.

Fig. 3 is a diagram illustrating an idealized cost-benefit model in accordance with an aspect of the present invention.

5 Fig. 4 is a schematic block diagram illustrating a multi-tiered question/answering system in accordance with an aspect of the present invention.

Figs. 5-7 illustrate various decision trees for determining the success and numbers of queries in accordance with an aspect of the present invention.

10 Figs. 8 and 9 illustrate example cost-benefit models in accordance with an aspect of the present invention.

Fig. 10 is a flow diagram illustrating a normalization process in accordance with an aspect of the present invention.

Fig. 11 is a schematic block diagram illustrating a suitable operating environment in accordance with an aspect of the present invention.

15 Fig. 12 is a schematic block diagram of a sample-computing environment with which the present invention can interact.

## DETAILED DESCRIPTION OF THE INVENTION

20 The present invention relates to a system and methodology to facilitate information extraction and learning from an unstructured corpora of information. This information can be retrieved from local and/or remote databases that house such information (*e.g.*, web sites, local databases, electronic encyclopedias or dictionaries). In one aspect of the present invention, a normalization system is provided. The normalization system includes an interface component that receives data corresponding to  
25 a heterogeneous knowledge base such as from web sites or other sources. A normalization component applies a statistical or logical model that relates the expected accuracy or quality of answers to sets of information-acquisition actions, and a utility model capturing the costs and benefits associated with information extractions, to provide a regularized understanding of the value of extracting information from the knowledge

base. The utility model is employed to provide dynamic controls for obtaining information relating to the costs of seeking the information versus the value of obtaining more accurate information. A preference tool or interface can be provided to aid in the assessment of respective costs and benefits for obtaining the information.

5           As used in this application, the terms “component,” “service,” “model,” “system,” and the like are intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. For example, a component may be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. 10 By way of illustration, both an application running on a server and the server can be a component. One or more components may reside within a process and/or thread of execution and a component may be localized on one computer and/or distributed between two or more computers.

          As used herein, the term “inference” refers generally to the process of reasoning 15 about or inferring states of the system, environment, and/or user from a set of observations as captured *via* events and/or data. Inference can be employed to identify a specific context or action, or can generate a probability distribution over states, for example. The inference can be probabilistic; that is, the computation of a probability distribution over states of interest based on a consideration of data and events. Inference 20 can also refer to logical techniques, such as rule-based policies, or compositional rules employed for composing higher-level events from a set of more atomic, lower-level events and/or data. Such inference results in the construction of new assessments, labels, assignments, or events or actions from a set of observed events and/or stored event data, whether or not the events are correlated in close temporal proximity, and whether the 25 events and data come from one or several event and data sources.

Referring initially to Fig. 1, a normalization system 100 employing a utility model for extracting information from an unstructured corpora is illustrated in accordance with an aspect of the present invention. An interface component 110 is provided that receives questions 120 posed by users. The questions 120 are processed in accordance with a



normalization component 130 that employs one or more predictive models that provide estimates of how different information extractions will enhance the accuracy of answers, and a utility model 140 that captures the costs and benefits of taking the information-extraction actions. The predictive models can also include an accuracy predictor 142 that can be employed in conjunction with the utility model 140. These components can be viewed as providing machinery that normalizes the system's understanding of how to interact with a potentially heterogeneous, ill-structured database 150 (also referred to as a knowledge base). The normalization process includes applying the utility model 140 and/or accuracy predictor 142 to dynamically control extraction of previously unknown or disassociated information from the database 150. For example, this can include reformulating the question 120 into an alternative set or subset of queries and controlling the number of queries submitted to the database 150 given various decision-theoretic considerations that are described in more detail below. It is noted that the database 150 can include closed system environments such as from a local databases, systems, files, directories, and/or sources such as an electronic encyclopedia, dictionary, or other information source. Also, the database 150 can include open system sources such as provided by remote web sites and/or other databases/sources (e.g., local/remote Intranet sites/databases associated with an entity).

In general, the utility model 140 applies a cost-benefit analysis to dynamically control the number and types of attempts made to acquire information or answers 160 from the database 150 in response to the question or questions 120. As will be described in more detail below, this includes an analysis of the costs of searching for information versus the benefits or value of obtaining more accurate answers to the questions 120. In addition to supplying answers 160, the interface component 110 may also initiate dialog with users at 160. Such dialog may be initiated by a dialog component (not shown) based upon predetermined probability thresholds or other criteria that can also include a cost-benefit analyses that considers when it would be best to ask a user to reformulate a question rather than expending effort on handling a query that may be expensive (in terms of searching for information from the knowledge base) or likely to yield inaccurate

results. In such an analysis, an assessment of the cost of delay and effort associated with a query reformulation and the likelihood that a reformulation would lead to a better result can be considered and modeled. Also, the dialog component can make a decision when to engage a user to request a reformulated question or additional information. This can include alerting or warning the user about the cost of receiving a good answer, or recommending that a query be attempted elsewhere (*e.g.*, a human expert). In order to determine respective costs and/or benefits, a preference specification and store 170 may be provided that enables users to assess or select various parameters that influence the utility model 140 which is described in more detail below with respect to Fig. 2.

Referring to Fig. 2, the relationship between a preference component 200 and utility model 210 is illustrated. The preference component 200 can be presented in the form of a Graphical User Interface to enable users to select and/or alter parameters that may affect the utility model 210. These inputs can include costs 220 that can be expressed in dollar amounts (or other meaningful parameter representing cost). Also, value amounts 230 can be expressed representing a parameter expressing the user's desire for accuracy in a respective answer. Other preferences may include language preferences 240 that indicate the type of languages the user desires to employ and thus, receive answers (*e.g.*, English, German, Spanish, and so forth). At 250, a user context component is provided that can include such inputs as a user's current location, velocity, actions with a computing device (*e.g.*, current speed of application switching, typing, recent history of a user's queries, recent text generated by user, *etc.*), calendar (*e.g.*, deadline coming up), as well as high-level inferences about a user's goals from multiple features, including those described below. Still other preference inputs 260 may be provided which are described in more detail below. After preferences have been established, the utility model 210 is then applied to normalize the knowledge base described above such as in controlling the numbers of reformulated queries that are submitted to search engines.

In controlling the number of queries relayed to a search engine, preferences about the costs of sending increasing numbers of queries and the benefits of a more accurate answer are determined. Several models for representing costs and benefits are feasible. In one example, a model where a user assesses a parameter  $v$ , indicating the dollar value of receiving a correct answer to a question, and where a parameter  $c$  represents the cost of each query rewrite submitted to a search engine can be employed. Rather than asserting a constant value for receiving an answer to a question, a user may consider the value of receiving an answer as a function of the details of the situation at hand. For example, the value of an answer may be linked to the type of question, goals, and the time of day for a user.

Similarly, the cost of submitting queries can be a function of such factors as the current load sensed on a search engine or the numbers of queries being submitted by a user's entire organization to a third-party search service, for example. The costs may also scale non-linearly with increasing numbers of queries. For example, the first  $n$  queries may be considered free by a search service supporting the question-answering systems at an enterprise, after which expenses are incurred in a supra-linear manner.

Models that output the probability of retrieving a successful answer, conditioned on different numbers of query rewrites, enable computing an expected value of submitting the queries. If the value of not receiving a valid answer is taken as zero, the expected value of submitting  $n$  queries is the product of the likelihood of the answer, given evidence  $E$  about the query,  $p(A|E, n, \xi)$ , and the value of obtaining a correct answer  $v$ ,  $p(A|E, n, \xi) v$ .

In one example, consider a preference model where the value of an answer,  $v$ , is assessed in terms of the cost of queries,  $c$ . That is, assess the value of answers as some multiple  $k$  of the cost of each query  $c$ ,  $v=kc$ . Also assume a cost model that grows linearly with the number of queries,  $nc$ . In making decisions about the ideal number of queries to submit, optimize a net expected value, computed as the difference of the expected value and cost, for different  $n$ . Thus, it is desired to find the ideal  $n$ ,

$$n^* = \arg \max_n p(A|E, n, \xi)kc - nc.$$

As noted above, question/answering systems can be extended with a utility model and/or models of accuracy (described below) as a function of the number of queries submitted, and these systems can check each quantity of query rewrites explored in machine learning studies to identify an optimized number of queries to submit.

5            Fig. 3 is a diagram 300 illustrating an idealized view of a cost-benefit analysis where the probability of an answer grows with decreasing marginal returns with additional queries. The expected value, cost, and net expected value are displayed as a function of the number of queries submitted. If there were smooth decreasing marginal returns on accuracy with increasing numbers of queries,  $n^*$  could be identified from the derivatives of the curves. As indicated in Fig. 3, the ideal number of queries to issue is  
10            obtained at a position on the  $x$ -axis where the change in expected value of the answer is equal to the cost of each query. In reality, given potential non-monotonicity of the expected value curve, the number of queries associated with each learned model is evaluated.

15            Fig. 4 illustrates a multi-tiered question/answering system in accordance with an aspect of the present invention. A rewriting component 410 receives questions 420 and submits the questions in reformulated terms described below to a ranking component 430. The ranking component 430 employs a first tier of learning models to order reformulated queries 440 in an order of likelihood of providing a suitable answer. The  
20            models can be trained from a training set of data that is described in more detail below.

              When the reformulated queries have been ordered according to the probability of obtaining a correct answer, a utility model and accuracy predictor 450 such as described above is employed to form a query subset 460 which is then submitted to one or more search engines 470. Results returned from the search engines 470 are processed by an  
25            answer composer 480 that generates a formatted answer or answers 490 in response to the question 420. A request formulation 492 may also be submitted to request that a user reformulate a question 420 (or try searching somewhere else *e.g.*, ask a human expert), or to warn the user about the cost of continuing. At 498, system feedback may be employed that forms a multistage analysis that takes results from a previous cycle of analysis into

consideration. Also, one or more properties 496 from the questions 420 may be analyzed by the utility model and accuracy predictor 450 when determining the query subset 460. The following discussion describes one possible implementation for the system 400. It is to be appreciated that various implementations are possible (*e.g.*, various components combined into single component, some components operating on remote systems, and so forth).

The system 400 reformulates user questions 420 or queries into likely substrings of declarative answers to the question. For each query, several rewrites are generated using eight rewrite heuristics (or other number). The rewrites vary from specific string matching to a simple "ANDing" of the query words. As an example, for the query "*Who killed Abraham Lincoln?*" there can be three rewrites: <LEFT> "killed Abraham Lincoln"; "Abraham Lincoln was killed by" <RIGHT>; and who AND killed AND Abraham AND Lincoln. <LEFT> and <RIGHT> refer to the likely placement of candidate answers.

The first two rewrites generally require that a text on the Web or knowledge base match the exact phrase, such as "killed Abraham Lincoln." The last rewrite is referred to as a conjunctive *back-off strategy*, as it "ANDs" together all the query words, leading to less specific queries. The rewrite strings are formulated as search engine queries and sent to a search engine from which page summaries are collected. Any search engine can be used as the provider of results to a second stage of analysis. The system 400 can assign heuristic scores to results of different types of rewrites. For example, the system assigns higher weights to results of more precise rewrites than it does to the more general back-off rewrite. Several phases of analysis can be employed to identify answers to questions from the results returned by searches with query rewrites and may include one or more of the following:

*Mine N-Grams.* From page summaries returned for query rewrites, all unigram, bigram and trigram word sequences are extracted. The *n*-grams are scored according to their frequency of occurrence and the weight of the query rewrite that retrieved it. As an example, the common *n*-grams for the example query about the assassination of Abraham

Lincoln are: Booth, Wilkes, Wilkes Booth, John Wilkes Booth, bullet, actor, president, Ford's, Gettysburg Address, derringer, assignation, *etc.*

*Filter N-Grams.* The  $n$ -grams are filtered and re-weighted according to how well each candidate matches the expected answer type, as specified by fifteen handwritten filters (or other number). These filters utilize surface-level string features, such as capitalization or the presence of digits. For example, for *When* or *How many* questions, answer strings with numbers are given higher weight, and for *Who* questions, answer strings with capitals are given added weight and those with dates are demoted.

*Tile N-Grams.* The  $n$ -grams are *tiled* together by lining up matching sub-phrases where appropriate, so that longer answers can be assembled from shorter ones. Following tiling, the answers to the example query are: John Wilkes Booth, bullet, president, actor, Ford. John Wilkes Booth receives a higher score than the other answer candidates because it is found in matches to specific rewrites and because it occurs often overall.

To limit the number of queries issued by the system 400, the expert-derived heuristic functions described above are replaced with Bayesian models that can generate probabilities of answer success. In an initial phase of analysis, models are employed that provide a ranking of individual queries. Bayesian learning procedures can be employed to generate models that can infer the probabilistic lift that queries of different types can yield the likelihood of an accurate answer. Such models provide a normalized metric for ordering sets of queries by their goodness or suitability, providing a decision surface for deliberating about the costs and benefits in a more global analysis of the end-to-end performance of the overall system.

Queries are generally separated into two categories: (1) queries that involve ANDing of individual words and occasionally short phrases (*e.g.*, population AND “of Japan”), and (2) queries that contain a single phrase (*e.g.*, “the population of Japan is”). The former is referred to as *conjunctural rewrites*. The latter is referred to as *phrasal rewrites*. These two sets of queries are associated with distinct evidential features, which are considered during modeling. For both types of rewrites, such features as the number

of distinct words and the number and percentage of stop words present in the queries are considered. For building predictive models of the goodness of *phrasal rewrites* similar features were also examined, but also included features derived from a statistical natural language parser for English text.

5           The syntactic parser constructs multiple parse trees, capturing multiple hypotheses for an input string, based on a consideration of the likely different parts of speech that words in a phrase can have. After producing all hypotheses, the parser employs a language model to rank the likely syntactic hypothesis, computing probabilities of each parse tree as the product of the probability of all of the nodes in the tree. Several features  
10       output by the parser were considered including the number of primary and secondary parses and the maximum probability parse tree, or a measure of grammatical “goodness” of a query rewrite. A list of the features used for both sets of query rewrites is listed in Tables 1 and 2.

15

Table 1: Features of conjunctional and phrasal rewrites considered in learning models of query goodness.

LONGPHRASE: The longest phrase in the rewrite, in terms of words.
LONGWD: The length of the longest word in the entire query.
NUMCAP: The number of capitalized words in the entire query.
NUMPHRASES: The total number of phrases in the overall query.
NUMSTOP: The number of stopwords in the entire query, using our list.
NUMWORDS: The number of words in the entire query string.
PCTSTOP: Percentage of stop words.

20

Table 2. Features used only for phrasal rewrites considered in learning models.

NUMCAP, NUMSTOP, PCTSTOP: as above.
PRIMARY_PASES: The number of primary parses given by the natural language parser.
SECONDARY_PASES: The number of secondary parses given by the natural language parser.
SGM: The “statistical goodness” of the rewrite; a measure of how grammatical the sentence or phrase is, given by the parser.

5           A Bayesian-network learning tool, named the WinMine was employed  
 (Chickering *et al.* in a publicly available paper on the Internet or other sources entitled “A  
 Bayesian Approach to Learning Bayesian Networks with Local Structure” (MSR-TR-97-  
 07, August 1997) to learn Bayesian networks and decision trees for the query rewrites.  
 To generate a set of queries for building models of query goodness, the system 400 was  
 10       executed on questions included in a training data set. This data set includes a set of  
 questions and correct answers used for evaluating the performance of question-answering  
 systems (*e.g.*, Trec9 data set).

It is noted that other types of features, modeling, and analysis may be employed in  
 accordance with determining answers and/or providing information to users. For  
 15       example, the use of full text and/or text summaries (short snippets) of articles may be  
 analyzed and returned by a knowledge base (*e.g.*, search engine). Also, automated  
 processes can include the process of learning logical or statistical predictive models that  
 predict the accuracy or quality of answers as a function of the nature or number of queries  
 issued to a knowledge base. Similar to above, Bayesian learning procedures can be  
 20       employed to learn the models. Furthermore, other features can be considered, including  
 higher-level features such as the distribution of topics associated with the results of  
 queries (*e.g.*, as can be identified with a statistical classifier that assigns topics based on  
 text being analyzed), and tags derived from natural-language parses of the initial  
 questions, and the text of results or the text of snippets returned from results, as well as



higher-level informational goals of the user, as derived from assessing goals directly, or as inferred from an analysis (*e.g.*, NLP analysis) of the user's initial question.

It is also noted that features can be analyzed in terms of classes of features, including attributes and statistics of attributes of morphological or semantic aspects of  
 5 initial (1) question and/or (2) one or more query results, including words and phrases, parts of speech, structure of natural language parse, length, topics and distribution of topics, and inferred or assessed informational goals or intentions.

Fig. 5 displays a decision tree 500 derived from a Bayesian model that maps properties of conjunctive rewrites to an expected accuracy. Fig. 6 depicts a model 600  
 10 for phrasal rewrites. These models provide a probabilistic score for judging the value of specific rewrites towards attaining a correct answer. The scores are heuristic in that the system does not employ single queries in normal operation, but utilizes ensembles of queries.

After an initial analysis, yielding models of the usefulness of individual rewrites,  
 15 a second phase of analyses, employed machine learning to build Bayesian models of the relationship between the ultimate accuracy of processing of questions and the numbers of queries submitted to a search engine, considering the properties of the question at hand. Such models enable cost-benefit analyses, trading off the expected gains in accuracy of an answer and the costs of additional queries. These analyses provide making dynamic  
 20 decisions about the number of queries to submit to a search service—and to make decisions about when to forego an analysis and, instead, to ask a user to reformulate their question. An ensemble of models was constructed by generating cases via a process of running the system on sample questions and applying different fixed thresholds on the number of rewrites submitted to search engines, as ordered by the goodness of queries  
 25 established in the first phase of model construction. Additional features employed in model construction are illustrated in Table 3. It is noted that these thresholds did not always result in the total number of rewrites being submitted because some questions generated fewer rewrites than the threshold values would allow.

Table 3. Features considered by the models for choosing rewrite thresholds for a given question-answering run.

AVERAGE_SNIPPETS_PER_REWRITE: Snippets are the summaries collected from web pages for a given query.
DIFF_SCORES_1_2: The difference between the first and second highest scored answer from AskMSR's scoring heuristic.
FILTER: The filter applied to the original query, such as "nlpwin_who_filter".
FILTER2 : These are very particular filters that focus on words and bigrams...
MAXRULE: Scores are given at the reformulation stage, based on the filter used to generate rewrites. This is the highest score procured for a particular query.
NUMNGRAMS: The overall number of ngrams mined from snippets.
RULESCORE_X: Number of ngrams for rules with score X.
STD_DEVIATION_ANSWER_SCORES: The std. deviation amongst the top five answer scores from AskMSR's heuristic.
TOTALQUERIES: Total queries issued after all rewrites.
TOTNONBAGSNIPS: Total snippets generated from <i>phrasal</i> rewrites.
TOTSNIPS: Total snippets for all rewrites.

5                   The number of queries were discretized into fixed thresholds at 1-10, 12, 15, and 20 rewrites per question, thus building thirteen models (other numbers can be employed). The models generated by this process provide predictions about the overall accuracy of answers to questions at increasingly higher levels of thresholds on query rewrites submitted to a back-end search engine. Fig. 7 displays a decision tree 700 learned from

10                   data about the performance of question answering when limiting submitted queries to 10 rewrites.

The following describes results of experiments that were conducted to measure performance of the normalization components described above. A set of experiments

were performed on the systems described above. Given a query, query rewrites are first sorted into a list by single-query models. Then, an ensemble of Bayesian models for different numbers of rewrites are employed in conjunction with a utility model to select the best number of rewrites to issue to a search engine (or engines). The search results  
 5 are then passed to an answer composition stage. The available actions are defined by the end-to-end performance models which were trained for 1-10, 12, 15, and 20 rewrites.

Fig. 8 shows a cost-benefit analysis graphically for an example query “Where is the Orinoco River?” with a cost per query of 1 and a correct answer valued at 10. In this case, the best decision available is to choose to submit 5 query rewrites. Fig. 9 displays  
 10 cost-benefit analysis for a query, “What currency does Argentina use?” for the same preference settings. With this policy, it is best to send 2 query rewrites to the search engine. Table 4 below shows the performance of the system with cost-benefit policies over different baseline policies. In these fixed-cost runs, the system is given a ceiling on the number of query rewrites it can use.

15 In a first set of experiments, the system chooses randomly from rewrites available for each query up to a threshold ( $N$ ). In a second set of experiments, the system was executed with a static policy of selecting  $N$  rewrites from a list of query rewrites, ranked by a probabilistic query-quality score described above. A ceiling of 20 rewrites is roughly equal to the policy in a legacy system, which had no limitation on rewrites, as  
 20 only a few queries yield more than 20 rewrites. As highlighted in the table of results, sequencing queries by the query-quality score dominates the randomly ordered queries, demonstrating the value of using the query-quality score.

Ranked query rewrites for cost-benefit analysis were also employed. Table 5 compares the policy chosen by the cost-benefit analysis with two fixed policies, one  
 25 using only conjunctive rewrites (top row) and the other using all rewrites (bottom row). These results show good performance for the system using the cost-benefit control (middle row). With the cost-benefit analysis, the system answers nearly as many correct as the original, unbounded system (277 versus 283), while posing less than a third of the total queries used without control.

As a baseline comparison, the system was also executed with a fixed policy of using only the conjunctive rewrite for each question (first row, Table 5). This is useful because the conjunctive rewrite is the query reformulation that nearly always leads to the most results from the search-engine backend. This makes the conjunctive rewrite extremely valuable, as a greater set of intermediate results means a better chance of finding an answer. This experiment shows that the conjunctive-query-only policy does fairly well, leading to 49% accuracy for only 499 total queries. However, this static policy is outperformed by the utility-directed system by a significant margin in terms of accuracy. Using the decision model, a 12% increase is achieved in correct answers at a cost of 680 additional queries.

Table 4. Cost and accuracy for a fixed rewrite policy compared with dynamic cost-benefit analysis.

Max Rewrites Per Question ( $N$ )	Total Cost	Correct Answers, Random Order	Correct Answers, Likelihood Order
$N = 1$	499	156	225
$N = 2$	946	217	238
$N = 3$	1383	243	254
$N = 4$	1805	252	278
$N = 5$	2186	272	282
$N = 6$	2490	268	282
$N = 7$	2738	272	282
$N = 8$	2951	279	282
$N = 9$	3103	276	282
$N = 10$	3215	281	282
$N = 12$	3334	281	283
$N = 15$	3410	282	283
$N = 20$	3426	283	283

Table 5. Cost (total queries) and accuracy with cost—benefit decisions for four different values for the value of an answer.

Rewrite Policy	Cost	Correct Answers (out of 499)
Conjunctural rewrites only	499	247
Cost-benefit $k=10, c=1$	1179	277
All rewrites	3426	283

5

Table 6 shows cost-benefit relationships for four different values of  $k$ . With a value of  $k=15$ , the performance of current systems is achieved but with many fewer queries (1346 vs. 3426).

10

Value of answer ( $k$ )	Cost	Correct answers
5	603	253
10	1179	277
15	1346	283
20	1405	283

15

Fig. 10 illustrates a normalization process 1000 in accordance with the present invention. While, for purposes of simplicity of explanation, the methodology is shown and described as a series of acts, it is to be understood and appreciated that the present invention is not limited by the order of acts, as some acts may, in accordance with the present invention, occur in different orders and/or concurrently with other acts from that shown and described herein. For example, those skilled in the art will understand and appreciate that a methodology could alternatively be represented as a series of interrelated states or events, such as in a state diagram. Moreover, not all illustrated acts may be required to implement a methodology in accordance with the present invention.

20

Proceeding to 1010, one or more questions posed by a user are received and processed. At 1020, the received questions are automatically reformulated or rewritten into a set of possible candidate queries. At 1030, user preferences are considered regarding the costs that are acceptable for searching for an answer along with respective value parameters relating to the accuracy of an answer. As noted above, such costs can be expressed in terms of a dollar amount or other parameter. Also, other inputs can be considered such as user goals in performing the search, time of day, language preferences and so forth. At 1040, a cost-benefit analysis is performed to determine an optimized number of queries from the reformulated queries that can be processed as a query subset from the reformulated queries. As noted above, the cost-benefit analysis can be employed to normalize information from a heterogeneous knowledge base. At 1050, the query subset is submitted to one or more search engines and/or databases. At 1060, when results are returned from the search engines, an answer is composed for the user question based upon the normalized request of information posed by the query subset

With reference to Fig.11, an exemplary environment 1110 for implementing various aspects of the invention includes a computer 1112. The computer 1112 includes a processing unit 1114, a system memory 1116, and a system bus 1118. The system bus 1118 couples system components including, but not limited to, the system memory 1116 to the processing unit 1114. The processing unit 1114 can be any of various available processors. Dual microprocessors and other multiprocessor architectures also can be employed as the processing unit 1114.

The system bus 1118 can be any of several types of bus structure(s) including the memory bus or memory controller, a peripheral bus or external bus, and/or a local bus using any variety of available bus architectures including, but not limited to, 11-bit bus, Industrial Standard Architecture (ISA), Micro-Channel Architecture (MSA), Extended ISA (EISA), Intelligent Drive Electronics (IDE), VESA Local Bus (VLB), Peripheral Component Interconnect (PCI), Universal Serial Bus (USB), Advanced Graphics Port (AGP), Personal Computer Memory Card International Association bus (PCMCIA), and Small Computer Systems Interface (SCSI).

The system memory 1116 includes volatile memory 1120 and nonvolatile memory 1122. The basic input/output system (BIOS), containing the basic routines to transfer information between elements within the computer 1112, such as during start-up, is stored in nonvolatile memory 1122. By way of illustration, and not limitation, nonvolatile memory 1122 can include read only memory (ROM), programmable ROM (PROM), electrically programmable ROM (EPROM), electrically erasable ROM (EEPROM), or flash memory. Volatile memory 1120 includes random access memory (RAM), which acts as external cache memory. By way of illustration and not limitation, RAM is available in many forms such as synchronous RAM (SRAM), dynamic RAM (DRAM), synchronous DRAM (SDRAM), double data rate SDRAM (DDR SDRAM), enhanced SDRAM (ESDRAM), Synchlink DRAM (SLDRAM), and direct Rambus RAM (DRRAM).

Computer 1112 also includes removable/non-removable, volatile/non-volatile computer storage media. Fig. 11 illustrates, for example a disk storage 1124. Disk storage 1124 includes, but is not limited to, devices like a magnetic disk drive, floppy disk drive, tape drive, Jaz drive, Zip drive, LS-100 drive, flash memory card, or memory stick. In addition, disk storage 1124 can include storage media separately or in combination with other storage media including, but not limited to, an optical disk drive such as a compact disk ROM device (CD-ROM), CD recordable drive (CD-R Drive), CD rewritable drive (CD-RW Drive) or a digital versatile disk ROM drive (DVD-ROM). To facilitate connection of the disk storage devices 1124 to the system bus 1118, a removable or non-removable interface is typically used such as interface 1126.

It is to be appreciated that Fig 11 describes software that acts as an intermediary between users and the basic computer resources described in suitable operating environment 1110. Such software includes an operating system 1128. Operating system 1128, which can be stored on disk storage 1124, acts to control and allocate resources of the computer system 1112. System applications 1130 take advantage of the management of resources by operating system 1128 through program modules 1132 and program data 1134 stored either in system memory 1116 or on disk storage 1124. It is to be

appreciated that the present invention can be implemented with various operating systems or combinations of operating systems.

5 A user enters commands or information into the computer 1112 through input device(s) 1136. Input devices 1136 include, but are not limited to, a pointing device such as a mouse, trackball, stylus, touch pad, keyboard, microphone, joystick, game pad, satellite dish, scanner, TV tuner card, digital camera, digital video camera, web camera, and the like. These and other input devices connect to the processing unit 1114 through the system bus 1118 *via* interface port(s) 1138. Interface port(s) 1138 include, for example, a serial port, a parallel port, a game port, and a universal serial bus (USB).

10 Output device(s) 1140 use some of the same type of ports as input device(s) 1136. Thus, for example, a USB port may be used to provide input to computer 1112, and to output information from computer 1112 to an output device 1140. Output adapter 1142 is provided to illustrate that there are some output devices 1140 like monitors, speakers, and printers, among other output devices 1140 that require special adapters. The output

15 adapters 1142 include, by way of illustration and not limitation, video and sound cards that provide a means of connection between the output device 1140 and the system bus 1118. It should be noted that other devices and/or systems of devices provide both input and output capabilities such as remote computer(s) 1144.

Computer 1112 can operate in a networked environment using logical connections

20 to one or more remote computers, such as remote computer(s) 1144. The remote computer(s) 1144 can be a personal computer, a server, a router, a network PC, a workstation, a microprocessor based appliance, a peer device or other common network node and the like, and typically includes many or all of the elements described relative to computer 1112. For purposes of brevity, only a memory storage device 1146 is

25 illustrated with remote computer(s) 1144. Remote computer(s) 1144 is logically connected to computer 1112 through a network interface 1148 and then physically connected *via* communication connection 1150. Network interface 1148 encompasses communication networks such as local-area networks (LAN) and wide-area networks (WAN). LAN technologies include Fiber Distributed Data Interface (FDDI), Copper



Distributed Data Interface (CDDI), Ethernet/IEEE 1102.3, Token Ring/IEEE 1102.5 and the like. WAN technologies include, but are not limited to, point-to-point links, circuit switching networks like Integrated Services Digital Networks (ISDN) and variations thereon, packet switching networks, and Digital Subscriber Lines (DSL).

5           Communication connection(s) 1150 refers to the hardware/software employed to connect the network interface 1148 to the bus 1118. While communication connection 1150 is shown for illustrative clarity inside computer 1112, it can also be external to computer 1112. The hardware/software necessary for connection to the network interface 1148 includes, for exemplary purposes only, internal and external technologies such as,  
10           modems including regular telephone grade modems, cable modems and DSL modems, ISDN adapters, and Ethernet cards.

          Fig. 12 is a schematic block diagram of a sample-computing environment 1200 with which the present invention can interact. The system 1200 includes one or more client(s) 1210. The client(s) 1210 can be hardware and/or software (*e.g.*, threads,  
15           processes, computing devices). The system 1200 also includes one or more server(s) 1230. The server(s) 1230 can also be hardware and/or software (*e.g.*, threads, processes, computing devices). The servers 1230 can house threads to perform transformations by employing the present invention, for example. One possible communication between a client 1210 and a server 1230 may be in the form of a data packet adapted to be  
20           transmitted between two or more computer processes. The system 1200 includes a communication framework 1250 that can be employed to facilitate communications between the client(s) 1210 and the server(s) 1230. The client(s) 1210 are operably connected to one or more client data store(s) 1260 that can be employed to store information local to the client(s) 1210. Similarly, the server(s) 1230 are operably  
25           connected to one or more server data store(s) 1240 that can be employed to store information local to the servers 1230.

          What has been described above includes examples of the present invention. It is, of course, not possible to describe every conceivable combination of components or methodologies for purposes of describing the present invention, but one of ordinary skill

in the art may recognize that many further combinations and permutations of the present invention are possible. Accordingly, the present invention is intended to embrace all such alterations, modifications and variations that fall within the spirit and scope of the appended claims. Furthermore, to the extent that the term “includes” is used in either the  
5 detailed description or the claims, such term is intended to be inclusive in a manner similar to the term “comprising” as “comprising” is interpreted when employed as a transitional word in a claim.